04

Ajustes de generación en jwt

Transcripción

[00:00] Ya tenemos nuestro token generado. Si venimos aquí, vas a ver que cuando le des a login, cuando entres al endpoint de login con tu credencial, este va a generar este token JWT, porque ahora está haciendo hardcodeado digamos con algunos valores por defecto que le estamos poniendo aquí. ¿Cuál es el objetivo ahora?

[00:22] Hacerlo dinámico porque no todos los usuarios van a tener el mismo subject, el mismo username. Entonces si lo quiero hacer dinámico, yo voy a crear aquí un parámetro del tipo usuario, porque quiero que cada token tenga los datos personalizados de cada usuario. Y aquí, por ejemplo en el subject lo que yo le voy a decir es que el usuario.getLlogin que me va a dar el username y también quiero devolver el id del usuario para mi cliente.

[00:50] Digamos que es un requerimiento del negocio que la aplicación cliente conozca el id de usuario. Aquí lo que yo puedo hacer es ponerle aquí .withClaim, y lo que yo voy a poner aquí es “id” seguido de usuario.getId() y listo. ¿Qué más puedo poner? En el token es muy común que tenga una fecha de validez, un tiempo de validez.

[01:23] Pueden ser dos horas, pueden ser dos minutos, puede ser un día entero, un año, no es recomendado, pero también hay. Entonces dependiendo en tiempo de validez, tú puedes ponerle que expire en cierta hora, por ejemplo aquí le voy a poner que expire en. Y aquí yo tengo que crear un instant.

[01:41] Para crear el instant, lo que yo le voy a decir aquí es crear un método que me retorne un instant, vamos a hacer un private Instante de Javatime, y aquí le voy a decir generarFechaExpiración. Y aquí lo que yo voy a retornar es un localDateTime porque yo le voy a agregar solamente dos horas de validación.

[02:21] Entonces vamos aquí. Le voy a decir que de un LocalDateTime de ahora, plus quiero que tenga más, plusHours(2).toInstant y de la zona que sea (ZoneOffset). Y vamos a ponerle un offset de Sudamérica que creo que sería de "-05:00" horas me parece. Vamos a ver.

[02:59] Y listo. Entonces cuando estoy, yo le voy a decir aquí que me genere la fecha de expiración. Si quieres hacerlo más dinámico tú le puedes enviar aquí el número como parámetro y automáticamente le puedes cargar dos, tres horas, eso ya depende de ti.

[03:18] Por ahora vamos a decirle que me genere esta fecha automáticamente porque el token va a expirar después de dos horas. Entonces, dependiendo del tiempo de hoy día más dos horas y listo. Y vamos a dejarlo ahí por ahora.

[03:33] Vamos a ir ahora a nuestro AutenticacionController porque tenemos que enviarle un parámetro del tipo usuario a este método, ¿cómo hacemos ahora? Lo que vamos a hacer es declarar en una variable aquí, var usuarioAutenticado. Este usuarioAutenticado va a salir si es que esta autenticación es exitosa.

[04:01] Vengo aquí y le voy a decir que de aquí me dé el principal, usuarioAutenticado.getPrincipal(). El principal viene a ser el objeto, el usuario que ya fue valga la redundancia, autenticado en nuestro sistema.

[04:13] Para que compile mi código lo que tengo que hacer aquí ahora es castearlo. Voy a castear a mi objeto usuario y ya no debería haber ningún problema. Yo le estoy diciendo que este usuarioAutenticado, este principal es igual a mí objeto usuario, que yo tengo en mi modelo.

[04:29] Esa sería la primera condición que yo tengo para mi generación de token. Entonces voy a probar aquí, voy a enviar, generó mi toquen y yo lo que quiero hacer ahora es verificar si la información que yo había agregado es visible y miren aquí. Vemos ahora que en efecto es el mismo subject, pero ahora tiene mi id de usuario.

[04:57] Por lo tanto sí lo obtuvo dinámicamente y la fecha de expiración, es el 22 de febrero a las 08:44, en este momento son las 06:44 entonces sí, en efecto son dos horas más que le ha dado de validación. Listo ya tenemos la primera parte de nuestras mejoras para nuestra generación de tokens listas. ¿Qué más nos falta?

[05:24] Yo no quiero recibir esto solo como un string, porque hasta ahora en nuestro API lo que estamos haciendo es consumir DTO y devolver DTO. Entonces para eso lo que yo voy hacer aquí es que me devuelva un new DatosJWTToken(). Es mi DTO que yo estoy creando. Y para esto yo voy a crear mi DTO que está aquí.

[05:53] Le voy a mandar como parámetro el JWTtoken y voy a crear este record. Y lo voy a crear dentro de mi paquete de seguridad, no lo voy a hacer ahora, voy a venir aquí, api.infra.security. Listo. Le damos okay. No lo voy a agregar aun, y automáticamente me está agregando la variable que le he mandado. Es un String y listo.

[06:26] Vengo aquí, guardo para ver cómo lo va a formatear ahora. Esperamos un poco que responda mi servidor. Listo y vamos a venir al token otra vez. Y ahora como pueden ver mi respuesta ya llega formateada en un JSON, ya viene mi JWTtoken correctamente formateado con el mismo token que yo retorné anteriormente, pero en formato más amigable.

[07:00] Siempre recuerden esto. Si en todo tu API y el estándar es consumir DTO y devolver DTO, eso debe aplicar para todos los métodos que estás usando. Es para mantener tu código limpio y mucho más entendible también para tu

aplicación cliente a veces consumir el string directamente no es tan fácil que digamos.

[07:19] Último punto que les quiero mostrar. Este token service tiene hardcodeado un secret y yo ya les comenté que esto es un anti-patrón, es una pésima práctica que no deberían hacer nunca porque hay casos en la vida real que se filtran credenciales, se filtran claves de acceso, incluso hasta para Amazon en la cual bueno, si tienes tus credenciales comprometidas, pueden comprometerte las cuentas.

[07:51] Es así como muchas veces los hackers se hacen de las credenciales de un usuario, ni siquiera las piden. Son los mismos errores, que uno comete cuando programa que las comparte sin darse cuenta. Lo que vamos a hacer ahora entonces es declarar un valor aquí en nuestro application.properties.

[08:11] Quizás esto es nuevo para ustedes, pero hasta ahora hemos sobrescrito propiedades propias de Spring para darle un comportamiento diferente al que viene por defecto en nuestro API. Yo aquí también puedo definir propiedades creadas por mí, por mi aplicación por eso son application.properties.

[08:32] Por ejemplo, lo que yo puedo crear aquí es decirle que de mi api.security el secret va a hacer igual a 123456. Entonces mi api.security.secret es del 1 al 6. Lo que voy a hacer ahora en mi token service es consumir ese valor de ese property. Para eso yo voy a crear aquí un String apiSecret; y este apiSecret va a darme ese valor aquí y listo.

[09:16] Con eso podría funcionar pero ahora ustedes me dirán: “Diego, pero este archivo de aquí también se comparte en el código fuente tú estás mandando el problema de este lado a este lado y estamos en lo mismo”. Y sí, si pensaron en eso felicidades también porque en efecto aquí no estamos solucionando nada. [09:39] Una de las técnicas para evitar hacer esto es usar variables de ambiente. ¿Qué son variables de ambiente? Son variables que existen en tu sistema. Por ejemplo ahorita yo estoy en mi computador, en mi

computador personal y yo no tengo ninguna variable de ambiente todavía personalizada expuesta.

[10:00] Pero digamos una vez que lo despliegues en el servidor de test, en producción, User accept, en cualquier ambiente que despliegues tu aplicación lo más probable es que tengas tus variables inyectadas ahí, sobre todo si estás usando Docker o Kubernetes, es muy fácil inyectarle variables en el container y esas variables pueden tener como valor el string que tú deseas que sea el secret.

[10:25] Por ejemplo, podría ahorita entrar a mi terminal, venir aquí y decirle que me exporte la variable JWT_SECRET=123456. Le doy export y si yo le hago un eco a esta variable JWT_SECRET, entonces me va a dar en 123456. Esa variable ya existe aquí en mi computador porque yo la acabo de crear.

[10:54] Lo mismo tú puedes hacer en tus otros servidores. Entonces de la misma forma que yo lo llamé acá, como la he exportado con este nombre, yo le voy a decir ahora que quiero la variable, se duplicó JWT_SECREAT. Porque esta es mi variable de ambiente que yo estoy usando. Voy a darle guardar.

[11:14] Voy a detener un rato mi servidor para que cargue la propiedad después nuevamente, voy a darle iniciar, vamos a esperar un poco. Mi aplicación ya inició y yo voy a llamar aquí mi servicio y vemos que me dio un internal server error 500. ¿Y cuál es el error que me está dando?

[11:40] Lo que me está diciendo es que el secret no puede ser null. ¿Esto por qué? Si bien yo he declarado aquí mi secret, yo en mi tokenService no he inicializado este string. Y si yo deseo obtener el valor de este string desde mi archivo de propiedades, lo que yo hago es copiar esto. Voy a copiar mi valor de la propiedad.

[12:08] Voy a tokenService y con la anotación, aquí no. Con la anotación @Value, yo voy entre comillas, abro signo de dólar, abro llaves y le voy a llevar aquí el valor de mi propiedad de la cual yo quiero extraer ese valor para mi

apiSecret. Voy a guardar nuevamente allí. Borro mi terminal y esperamos un momento.

[12:37] Como pueden ver entonces si yo declaro esta propiedad aquí, incluso el IDE ya me ayuda porque como ya está siendo usada, ya no está subrayado en amarillo. Simplemente él ya relacionó que va a ser usado aquí en esta parte de mi código. Voy a llamar nuevamente mi endpoint, voy a hacer otra vez el llamado porque no está conectando. Vamos a ver, a mí me parece que salió una excepción por aquí.

[13:04] ¿Qué es lo que dicen? Que no puede resolver el placeholder "JWT_Secret" en "${JWT_SECRET}". ¿Esto por qué? Porque quizás mi IDE, mi aplicación no tiene acceso a mis variables de entorno de mi sistema.

[13:18] Y cuando sucede eso por ejemplo, cuando tú no puedes obtener ese valor de tu variables de entorno, tú puedes definirle un valor por defecto, por ejemplo, 123456. De esta forma yo tengo la variable entorno de mi computador, pero como mi IDE no tiene acceso a esas variables de entorno, entonces lo que él me dice es: "Yo no sé qué variable es esta".

[13:42] Y eso te puede pasar también en algún ambiente de producción, en algún ambiente de test, es un error común. Para esos casos, por defecto en mi ambiente de desarrollo que es este de aquí, yo le estoy poniendo un valor por defecto, que es el mismo, en este caso voy a guardar, vamos a limpiar aquí.

[14:00] De modo que si no encuentra esto, por defecto va usar este valor de aquí. Y eso soluciona todo mi problema, esperamos que recargue. No sé si recargó. Ahora sí, recargó mi servidor. Entro aquí, vamos enviarlo y vemos que ya está funcionando nuevamente mi servicio con el valor obtenido directamente desde mi archivo de propiedades.

[14:26] De este modo digamos, que si yo tengo un JWT_SECRET exclusivo para producción y debería ser así el estándar, entonces ya no está compartido en mi

código y esto de aquí, simplemente es un valor por defecto para ser usado en mi entorno de desarrollo y mis pruebas.

[14:43] De esta forma concluimos la cuarta clase y su curso hemos visto ya como obtener, cómo implementar la librería de Auth0 para nuestro propio JWT. Hemos aprendido también a consumir el usuario autenticado de nuestro proceso de autenticación hecho por el AuthenticationManager de Spring.

[15:03] Sin más que decirles, practiquen mucho esto. Este tópico suele ser de los más solicitados en el mundo laboral si tú eres familiar con técnicas de autenticación o autorización de datos, entonces estudia bastante esto.

[15:20] En la siguiente clase vamos a culminar con lo que es autorización en sí, porque si venimos a nuestro cliente, ya generamos el token pero aún así por ejemplo si yo voy a listar mis médicos, le voy a dar clic y sigo accesando a mi listado de médicos. Yo necesito desde ahora decirle a todos mis otros endpoints, por si acaso si el request no lleva este token que es válido en el header, entonces recházalo, que eso ya sería la parte de autorización. Pero eso es tema de la clase número 5. Nos vemos.